

[illegible]

[illegible]

```
1 0001 0 MODULE RPG$DSPLY( %TITLE 'DSPLY an item'
2 0002 0 IDENT = '1-004' ! file RPGDISPLY.B32 EDIT:DG1004
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1 ++
30 0030 1
31 0031 1
32 0032 1 FACILITY: RPGII SUPPORT
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 This module contains the RTL routine that supports the RPG DSPLY
37 0037 1 opcode.
38 0038 1
39 0039 1 ENVIRONMENT: VAX/VMS user mode
40 0040 1
41 0041 1 AUTHOR: Debess Grabazs, CREATION DATE: 18-March-1983
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 1-001 - Original version. DG 18-Mar-1983.
46 0046 1 1-002 - use routine COB$$RET_A_AB_PREV to get address of COB$$AB_PREV.
47 0047 1 MDL 29-Aug-1983
48 0048 1 1-003 - Create temporary for overpunched numeric before calling
49 0049 1 COB$ACC_SCR. In case of invalid entry, original value will
50 0050 1 be saved. DG 30-Aug-1983.
51 0051 1 1-004 - Set RPG for calls to both COB$ACC_SCR and COB$DISP_SCR.
52 0052 1 DG 24-Oct-1983.
53 0053 1
54 0054 1 --
55 0055 1 <BLF/PAGE>
```

```
57 0056 1 %SBTTL 'Declarations'
58 0057 1 +
59 0058 1 | PROLOGUE FILE:
60 0059 1 | -
61 0060 1
62 0061 1 REQUIRE 'RTLIN:RPGPROLOG';
63 0126 1 | Switches, PSECTs, macros,
64 0127 1 | Linkages and LIBRARYs
65 0128 1 | +
66 0129 1 | LINKAGES
67 0130 1 | NONE
68 0131 1 | -
69 0132 1 | +
70 0133 1 |
71 0134 1 | TABLE OF CONTENTS:
72 0135 1 | -
73 0136 1
74 0137 1 FORWARD ROUTINE
75 0138 1 RPG$DSPLY: NOVALUE;
76 0139 1
77 0140 1 | +
78 0141 1 | INCLUDE FILES
79 0142 1 | NONE
80 0143 1 | -
81 0144 1
82 0145 1 | +
83 0146 1 | MACROS
84 0147 1 | NONE
85 0148 1 | -
86 0149 1
87 0150 1 | +
88 0151 1 | EQUATED SYMBOLS
89 0152 1 | NONE
90 0153 1 | -
91 0154 1
92 0155 1 | +
93 0156 1 | EXTERNAL REFERENCES
94 0157 1 | -
95 0158 1
96 0159 1 EXTERNAL ROUTINE
97 0160 1 COB$ACC_SCR,
98 0161 1 COB$DISP_SCR,
99 0162 1 COB$$RET_A_AB_PREV,
100 0163 1
101 0164 1 LIB$SIGNAL;
102 0165 1
103 0166 1 EXTERNAL LITERAL
104 0167 1 RPG$_INVNUMENT;
105 0168 1
106 0169 1 EXTERNAL
107 0170 1 RPG$BTZ;

! ACCEPT with conversion
! DISPLAY with conversion
! Return address of table giving
! call history for advancing purposes
! Signal warning and continue execution

! Invalid numeric entry warning

! Table for translate blank to zero
```

```
109 0171 1 %SBTTL 'RPG$DSPLY - DSPLY an item'
110 0172 1 GLOBAL ROUTINE RPG$DSPLY(
111 0173 1     FLAGS,
112 0174 1     DISPLAY_FIELD: REF BLOCK[,BYTE],
113 0175 1     ACCEPT_FIELD:  REF BLOCK[,BYTE]
114 0176 1 ): NOVALUE =
115 0177 1
116 0178 1 ++
117 0179 1
118 0180 1 FUNCTIONAL DESCRIPTION:
119 0181 1
120 0182 1     This routine supports the RPG DSPLY opcode.
121 0183 1
122 0184 1 CALLING SEQUENCE:
123 0185 1
124 0186 1     CALL RPG$DSPLY (flags.fl.v [,display_field.rx.dx]
125 0187 1                   [,accept_fie(d.mx.dx)])
126 0188 1
127 0189 1 FORMAL PARAMETERS:
128 0190 1
129 0191 1     flags                                bit 0 is set on if a comma should be
130 0192 1                                         used in place of decimal point; bit 1
131 0193 1                                         is set on if blanks in an overpunched
132 0194 1                                         numeric field should be treated as
133 0195 1                                         equivalent to zeroes.
134 0196 1
135 0197 1     display_field                        data item whose value is displayed.
136 0198 1
137 0199 1     accept_field                        data item whose value is displayed and
138 0200 1                                         a new value is accepted into.
139 0201 1
140 0202 1 IMPLICIT INPUTS:
141 0203 1
142 0204 1     NONE
143 0205 1
144 0206 1 IMPLICIT OUTPUTS:
145 0207 1
146 0208 1     NONE
147 0209 1
148 0210 1 ROUTINE VALUE:
149 0211 1
150 0212 1     NONE
151 0213 1
152 0214 1 SIDE EFFECTS:
153 0215 1
154 0216 1     NONE
155 0217 1
156 0218 1 --
```

```
158 0219 2 BEGIN
159 0220
160 0221 LITERAL
161 0222 TRUE = 1,
162 0223 DISP = 0,
163 0224
164 0225 DEC_IS_COMMA_BIT = 1,
165 0226 BTZ_BIT = 2,
166 0227 CONVERT_BIT = 32,
167 0228 COMMA_BIT = 64,
168 0229 PROTECT_BIT = 256,
169 0230 RPG_BIT = 2048;
170 0231
171 0232 LOCAL
172 0233 COB$$AB_PREV,
173 0234 DEV : WORD,
174 0235
175 0236
176 0237 FLAG,
177 0238 NRO FLAG : INITIAL (0),
178 0239 TEMP_NRO : BLOCK [12,BYTE] VOLATILE,
179 0240 TEMP_STRING : VECTOR [15,BYTE];
180 0241
181 0242
182 0243 BUILTIN
183 0244 ACTUALCOUNT;
184 0245
185 0246
186 0247
187 0248
188 0249 Get address of history of previous call.
189 0250 Note that using it this way will affect the whole longword
190 0251 at the address, and not just the first byte.
191 0252
192 0253 COB$$AB_PREV = COB$$RET_A_AB_PREV();
193 0254
194 0255
195 0256
196 0257 Set up FLAG parameter.
197 0258
198 0259
199 0260 FLAG = CONVERT_BIT + RPG_BIT; ! Conversion and RPG
200 0261 IF (.FLAGS AND DEC_IS_COMMA_BIT) NEQ 0 ! Decimal is comma
201 0262 THEN FLAG = .FLAG + COMMA_BIT;
202 0263
203 0264
204 0265
205 0266 Deal with DISPLAY_FIELD.
206 0267
207 0268
208 0269 IF .DISPLAY_FIELD NEQ 0
209 0270 THEN
210 0271 BEGIN
211 0272
212 0273 IF (.DISPLAY_FIELD[DSC$$B_DTYPE] EQL DSC$$K_DTYPE_NRO) AND
213 0274 ((.FLAGS AND BTZ_BIT) NEQ 0)
214 0275 THEN
```

```
215 0276 3
216 0277
217 0278
218 0279
219 0280
220 0281
221 0282
222 0283
223 0284
224 0285
225 0286
226 0287
227 0288
228 0289
229 0290
230 0291
231 0292
232 0293
233 0294
234 0295
235 0296
236 0297
237 0298
238 0299
239 0300
240 0301
241 0302
242 0303
243 0304
244 0305
245 0306
246 0307
247 0308
248 0309
249 0310
250 0311
251 0312
252 0313
253 0314
254 0315
255 0316
256 0317
257 0318
258 0319
259 0320
260 0321
261 0322
262 0323
263 0324
264 0325
265 0326
266 0327
267 0328
268 0329
269 0330
270 0331
271 0332 5

      +
      - Convert blanks to zeroes if flag is set.
      CH$TRANSLATE (RPG$BTZ,
        .DISPLAY_FIELD[DSC$W_LENGTH], .DISPLAY_FIELD[DSC$A_POINTER],
        0, .DISPLAY_FIELD[DSC$W_LENGTH], .DISPLAY_FIELD[DSC$A_POINTER]);

      .COB$$AB_PREV = DISP; ! Set up history for proper advancing
      +
      - Display the field.
      COB$DISP_SCR (1, .DISPLAY_FIELD, .FLAG);
      END;

      +
      - Deal with ACCEPT_FIELD.
      IF (ACTUALCOUNT() EQL 3 AND .ACCEPT_FIELD NEQ 0)
      THEN
      BEGIN
      IF (.ACCEPT_FIELD[DSC$B_DTYPE] EQL DSC$K_DTYPE_NRO) AND
      ((.FLAGS AND BTZ_BIT) NEQ 0)
      THEN
      +
      - Convert blanks to zeroes if flag is set.
      CH$TRANSLATE (RPG$BTZ,
        .ACCEPT_FIELD[DSC$W_LENGTH], .ACCEPT_FIELD[DSC$A_POINTER],
        0, .ACCEPT_FIELD[DSC$W_LENGTH], .ACCEPT_FIELD[DSC$A_POINTER]);

      .COB$$AB_PREV = DISP; ! Set up history for proper advancing
      +
      - Display the field.
      COB$DISP_SCR (1, .ACCEPT_FIELD, .FLAG);
      FLAG = .FLAG OR PROTECT_BIT; ! Protect
      WHILE TRUE DO
      +
      - Keep ACCEPTing until successful.
      BEGIN
      BIND
      ZERO = UPLIT ('0');

      .COB$$AB_PREV = DISP; ! Set up history for proper advancing
      IF .ACCEPT_FIELD [DSC$B_DTYPE] EQL DSC$K_DTYPE_NRO
      THEN
      NRO_FLAG = TRUE;
      IF .NRO_FLAG
      THEN
      BEGIN
```

```
272 0333 5
273 0334 5
274 0335 5
275 0336 5
276 0337 5
277 0338 5
278 0339 5
279 0340 5
280 0341 5
281 0342 5
282 0343 5
283 0344 6
284 0345 6
285 0346 6
286 0347 6
287 0348 6
288 0349 6
289 0350 6
290 0351 5
291 0352 5
292 0353 5
293 0354 5
294 0355 5
295 0356 5
296 0357 5
297 0358 5
298 0359 5
299 0360 5
300 0361 4
301 0362 4
302 0363 4
303 0364 4
304 0365 4
305 0366 6
306 0367 6
307 0368 5
308 0369 5
309 0370 4
310 0371 4
311 0372 5
312 0373 5
313 0374 5
314 0375 5
315 0376 5
316 0377 5
317 0378 5
318 0379 4
319 0380 4
320 0381 3
321 0382 3
322 0383 3
323 0384 3
324 0385 3
325 0386 3
326 0387 3
327 0388 3
328 0389 3

+ Overpunched numeric string must be saved because
COB$ACC_SCR zeroes it out and if there is invalid
input, the original string must be available in
case the user responds to the reprompt with a <CR>.
-
TEMP_NRO [DSC$W_LENGTH] = .ACCEPT_FIELD [DSC$W_LENGTH];
TEMP_NRO [DSC$B_DTYPE] = DSC$K_DTYPE_NRO;
TEMP_NRO [DSC$A_POINTER] = TEMP_STRING;
IF .ACCEPT_FIELD [DSC$B_CLASS] EQL DSC$K_CLASS_SD
THEN
    BEGIN
        TEMP_NRO [DSC$B_CLASS] = DSC$K_CLASS_SD;
        TEMP_NRO [DSC$B_SCALE] = .ACCEPT_FIELD [DSC$B_SCALE];
        TEMP_NRO [DSC$B_DIGITS] = .ACCEPT_FIELD [DSC$B_DIGITS];
    END
ELSE
    TEMP_NRO [DSC$B_CLASS] = DSC$K_CLASS_S;

+ Initialize the temporary string.
-
CH$MOVE (.ACCEPT_FIELD [DSC$W_LENGTH],
        .ACCEPT_FIELD [DSC$A_POINTER],
        TEMP_STRING);

END;

+ Accept into the field.
-
IF ( COB$ACC_SCR (0, (IF .NRO_FLAG
                    THEN TEMP_NRO
                    ELSE .ACCEPT_FIELD),
                    .FLAG, 0, 0, 0, 0) )
THEN EXITLOOP
ELSE
    BEGIN
        + Error - print warning and re-prompt.
        -
        LIB$SIGNAL (RPG$ INVNUMENT);
        .COB$$$AB_PREV = DISP;          ! Set up history for proper advancing
    END;

END;

IF .NRO_FLAG
THEN
    + Move accepted string from temporary to actual accept field.
    -
    CH$MOVE (.ACCEPT_FIELD [DSC$W_LENGTH], TEMP_STRING,
            .ACCEPT_FIELD [DSC$A_POINTER]);
```

RPG\$DSPLY  
1-004

DSPLY an item  
RPG\$DSPLY - DSPLY an item

K 12  
16-Sep-1984 02:13:33  
14-Sep-1984 13:04:17

VAX-11 Bliss-32 V4.0-742  
[RPGRTL.SRC]RPG\$DSPLY.B32;1

Page 7  
(4)

```
: 329
: 330
: 331
: 332

0390 3
0391 2
0392 2
0393 1

END;
END;
```

```
00 00 00 30 00000 P.AAA:
ZERO=
OFFC 00000
5B 00000000G 00 9E 00002
5A 00000000G 00 9E 00009
5E 1C C2 00010
58 D4 00013
00000000G 00 00 00015
59 50 0001C
57 0820 8F 3C 0001F
04 04 AC E9 00024
57 40 A7 9E 00028
56 08 AC D0 0002C 1$:
13 02 A6 91 00032
0D 12 00036
6C 21 E1 00038
09 00 04 B6 66 2E 0003C
04 B6 66 00042
7E 56 7D 00047 2$:
01 DD 0004A
6B 03 FB 0004C
03 6C 91 0004F 3$:
01 13 00052
04 00054
0C AC D5 00055 4$:
01 12 00058
04 0005A
56 0C AC D0 0005B 5$:
13 02 A6 91 0005F
0D 12 00063
09 00 04 6C 21 E1 00065
04 B6 66 2E 00069
B6 66 0006F
7E 56 7D 00074 6$:
01 DD 00077
6B 03 FB 00079

.TITLE RPG$DSPLY DSPLY an item
.IDENT \1-004\
.PSECT _RPG$CODE,NOWRT, SHR, PIC,2
.ASCII \0\<0><0><0>
;
.P.AAA
.EXTRN COB$ACC_SCR, COB$DISP_SCR
.EXTRN COB$$RET_A_AB_PREV
.EXTRN LIB$SIGNAL, RPG$_INVNUMENT
.EXTRN RPG$BTZ
.ENTRY RPG$DSPLY, Save R2,R3,R4,R5,R6,R7,R8,R9,-
R10,R11
MOVAB COB$DISP_SCR, R11
MOVAB RPG$BTZ, R10
SUBL2 #28, SP
CLRL NRO_FLAG
CALLS #0, COB$$RET_A_AB_PREV
MOVL R0, COB$$AB_PREV
MOVZWL #2080, FLAG
BLBC FLAG, 1$
MOVAB 64(R7), FLAG
MOVL DISPLAY_FIELD, R6
BEQL 3$
CMPB 2(R6), #19
BNEQ 2$
BBC #33, FLAG, 2$
MOVTC (R6), @4(R6), #0, RPG$BTZ, (R6), @4(R6)
CLRL (COB$$AB_PREV)
MOVQ R6, -(SP)
PUSHL #1
CALLS #3, COB$DISP_SCR
CMPB (AP), #3
BEQL 4$
RET
TSTL ACCEPT_FIELD
BNEQ 5$
RET
MOVL ACCEPT_FIELD, R6
CMPB 2(R6), #19
BNEQ 6$
BBC #33, FLAG, 6$
MOVTC (R6), @4(R6), #0, RPG$BTZ, (R6), @4(R6)
CLRL (COB$$AB_PREV)
MOVQ R6, -(SP)
PUSHL #1
CALLS #3, COB$DISP_SCR
```

RPG\$DSPLY  
1-004

DSPLY an item  
RPG\$DSPLY - DSPLY an item

L 12  
16-Sep-1984 02:13:33  
14-Sep-1984 13:04:17

VAX-11 Bliss-32 V4.0-742  
[RFGRTL.SRC]RPG\$DSPLY.B32;1

Page 8  
(4)

	57	0100	8F	A8	0007C	BISW2	#256, FLAG	0316
	13	02	69	D4	00081	CLRL	(COB\$AB_PREV)	0326
			A6	91	00083	CMPB	2(R6), #T9	0327
	58		03	12	00087	BNEQ	8\$	0329
	2B		01	D0	00089	MOVL	#1, NRO FLAG	0330
10	AE		58	E9	0008C	BLBC	NRO FLAG, 11\$	0339
12	AE		66	B0	0008F	MOVW	(R6), TEMP_NRO	0340
14	AE		13	90	00093	MOVAB	#19, TEMP_NRO+2	0341
	09	03	6E	9E	00097	MOVAB	TEMP_STRING, TEMP_NRO+4	0342
			A6	91	0009B	CMPB	3(R6), #9	0346
	13	AE	10	12	0009F	BNEQ	9\$	0347
	18	AE	09	90	000A1	MOVAB	#9, TEMP_NRO+3	0348
	19	AE	08	A6	90	MOVAB	8(R6), TEMP_NRO+8	0342
			09	A6	90	MOVAB	9(R6), TEMP_NRO+9	0352
	13	AE	04	11	000AF	BRB	10\$	0357
6E	04	B6	01	90	000B1	MOVAB	#1, TEMP_NRO+3	0366
			66	28	000B5	MOVW3	(R6), @4(R6), TEMP_STRING	0369
			7E	7C	000BA	CLRL	-(SP)	0366
			7E	7C	000BC	CLRL	-(SP)	0366
	08		57	DD	000BE	PUSHL	FLAG	0376
	50	24	58	E9	000C0	BLBC	NRO FLAG, 12\$	0377
			AE	9E	000C3	MOVAB	TEMP_NRO, R0	0383
			50	DD	000C7	PUSHL	R0	0389
			02	11	000C9	BRB	13\$	0393
			56	DD	000CB	PUSHL	R6	0394
			7E	D4	000CD	CLRL	-(SP)	0395
00000000G	00		07	FB	000CF	CALLS	#7, COB\$ACC_SCR	0396
	11		50	E8	000D6	BLBS	R0, 14\$	0397
00000000G	00	00000000G	8F	DD	000D9	PUSHL	#RPG\$ INVNUMENT	0398
			01	FB	000DF	CALLS	#1, LTB\$SIGNAL	0399
			69	D4	000E6	CLRL	(COB\$AB_PREV)	0400
			97	11	000E8	BRB	7\$	0401
	05		58	E9	000EA	BLBC	NRO FLAG, 15\$	0402
04	B6		66	28	000ED	MOVW3	(R6), TEMP_STRING, @4(R6)	0403
	6E		04	000F2	15\$:	RET		0404

; Routine Size: 243 bytes, Routine Base: \_RPG\$CODE + 0004

: 333 0394 1  
: 334 0395 0 END ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
_RPG\$CODE	247	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

RPG\$DSPLY  
1-004

DSPLY an item  
RPG\$DSPLY - DSDPLY an item

M 12  
16-Sep-1984 02:13:33  
14-Sep-1984 13:04:17

VAX-11 Bliss-32 V4.0-742  
[RPGRTL.SRC]RPGDSPLY.B32;1

Page 9  
(4)

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	9	0	581	00:01.0
-\$255\$DUA28:[RPGRTL.OBJ]RPGLIB.L32;1	54	0	0	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:RPGDSPLY/OBJ=OBJ\$:RPGDSPLY MSRC\$:RPGDSPLY/UPDATE=(ENH\$:RPGDSPLY)

: Size: 243 code + 4 data bytes  
: Run Time: 00:07.3  
: Elapsed Time: 00:25.5  
: Lines/CPU Min: 3259  
: Lexemes/CPU-Min: 14079  
: Memory Used: 103 pages  
: Compilation Complete

0331

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY